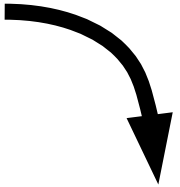
Machine Learning with Scikit-Learn

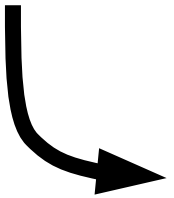
Andreas Mueller (NYU Center for Data Science, scikit-learn)

Me


universitätbonn
ais




Amazon
Machine
Learning




NYU
Center for
Data Science

Classification
Regression
Clustering
Semi-Supervised Learning
Feature Selection
Feature Extraction
Manifold Learning
Dimensionality Reduction
Kernel Approximation
Hyperparameter Optimization
Evaluation Metrics
Out-of-core learning
……


agramfort
Alexandre Gramfort


AlexanderFabisch
Alexander Fabisch


alextp
Alexandre Passos


amueller
Andreas Mueller


arjoly
Arnaud Joly


bdholt1
Brian Holt


bthirion
bthirion


chrisfilo
Chris Filo Gorgole...


cournape
David Cournapeau


duchesnay
Duchesnay


dwf
David Warde-Farley


fabianp
Fabian Pedregosa


GaelVaroquaux
Gael Varoquaux


glouppe
Gilles Louppe


jakevdp
Jake Vanderplas


jaquesgrobler
Jaques Grobler


jnothman


kastnerkyle
Kyle Kastner


kuantkid
Wei LI


larsmans
Lars


lucidfrontier45
Shiqiao Du


mblondel
Mathieu Blondel


MechCoder
Manoj Kumar


ndawe
Noel Dawe


NelleV
Varoquaux


ogrisel
Olivier Grisel


paolo-losi
Paolo Losi


pprett
Peter Prettenhofer


robertlayton
Robert Layton


ronw
Ron Weiss


satra
Satrajit Ghosh


sklearn-ci


vene
Vlad Niculae


VirgileFritsch
Virgile Fritsch


vmichel
Vincent Michel


yarikoptic
Yaroslav Halchenko

Home Installation Documentation Examples

Google™ Custom Search Search ×

Documentation of scikit-learn 0.17

Quick Start

A very short introduction into machine learning problems and how to solve them using scikit-learn. Introduced basic concepts and conventions.

User Guide

The main documentation. This contains an in-depth description of all algorithms and how to apply them.

Other Versions

- scikit-learn 0.18 (development)
- scikit-learn 0.17 (stable)
- scikit-learn 0.16
- scikit-learn 0.15

Tutorials

Useful tutorials for developing a feel for some of scikit-learn's applications in the machine learning field.

API

The exact API of all functions and classes, as given by the docstrings. The API documents expected types and allowed features for all functions, and all parameters available for the algorithms.

Additional Resources

Talks given, slide-sets and other information relevant to scikit-learn.

Contributing

Information on how to contribute. This also contains useful information for advanced users, for example how to build their own estimators.

Flow Chart

A graphical overview of basic areas of machine learning, and guidance which kind of algorithms to use in a given situation.

FAQ

Frequently asked questions about the project and contributing.

http://scikit-learn.org/

Doing Machine Learning With Scikit-Learn

Representing Data

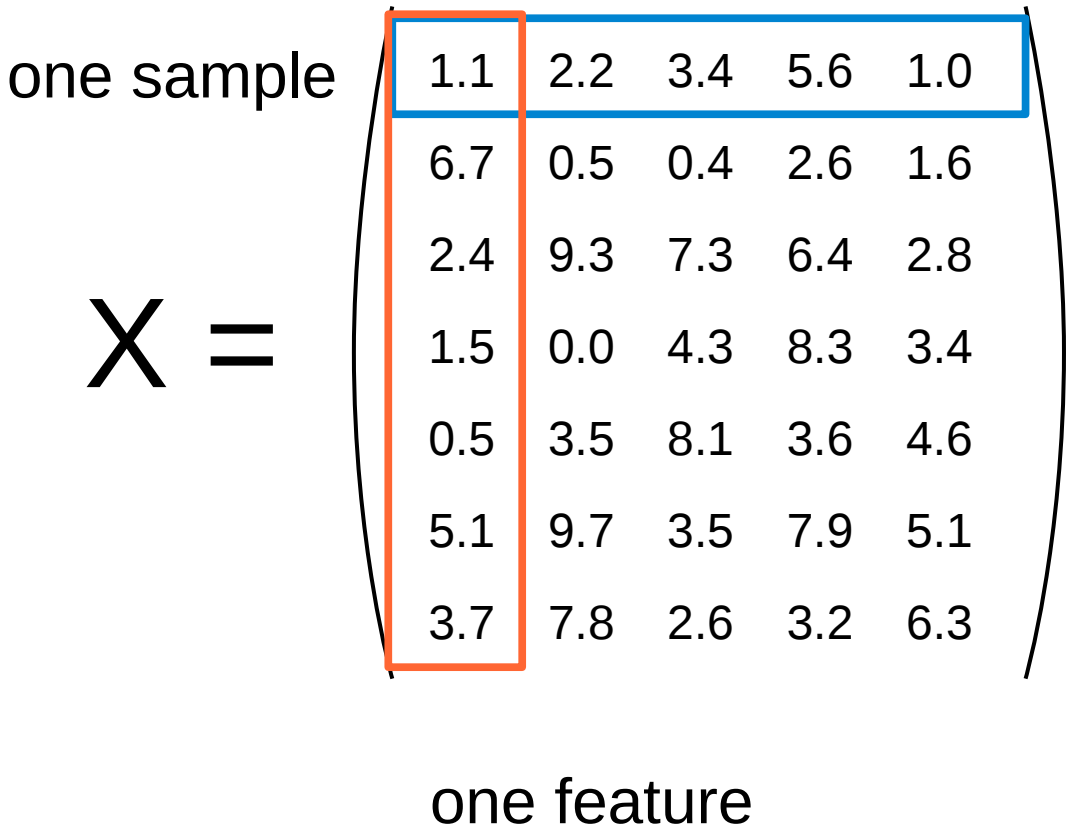
$$X = \begin{pmatrix} 1.1 & 2.2 & 3.4 & 5.6 & 1.0 \\ 6.7 & 0.5 & 0.4 & 2.6 & 1.6 \\ 2.4 & 9.3 & 7.3 & 6.4 & 2.8 \\ 1.5 & 0.0 & 4.3 & 8.3 & 3.4 \\ 0.5 & 3.5 & 8.1 & 3.6 & 4.6 \\ 5.1 & 9.7 & 3.5 & 7.9 & 5.1 \\ 3.7 & 7.8 & 2.6 & 3.2 & 6.3 \end{pmatrix}$$

Representing Data

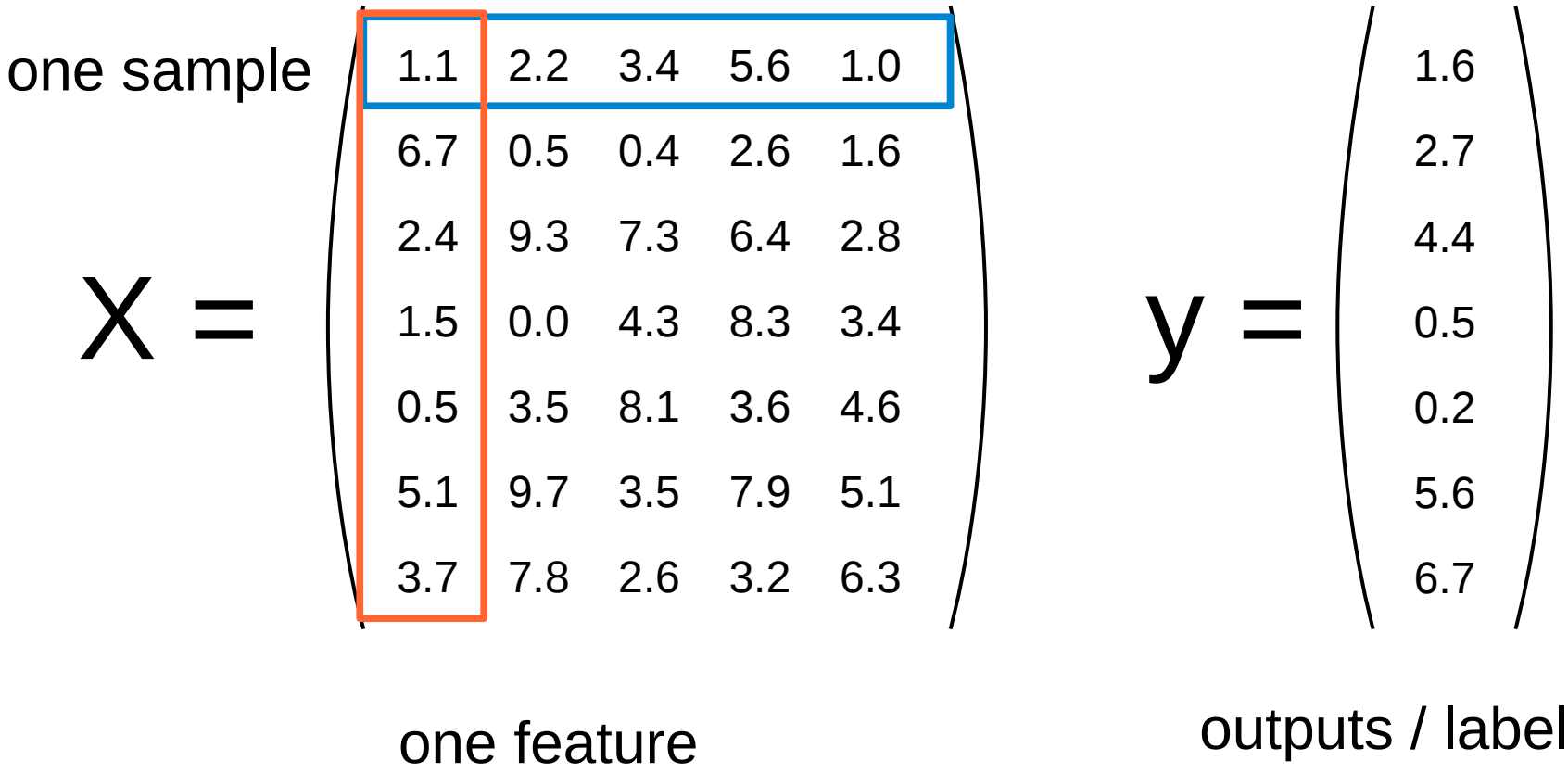
one sample

$$X = \begin{pmatrix} 1.1 & 2.2 & 3.4 & 5.6 & 1.0 \\ 6.7 & 0.5 & 0.4 & 2.6 & 1.6 \\ 2.4 & 9.3 & 7.3 & 6.4 & 2.8 \\ 1.5 & 0.0 & 4.3 & 8.3 & 3.4 \\ 0.5 & 3.5 & 8.1 & 3.6 & 4.6 \\ 5.1 & 9.7 & 3.5 & 7.9 & 5.1 \\ 3.7 & 7.8 & 2.6 & 3.2 & 6.3 \end{pmatrix}$$

Representing Data

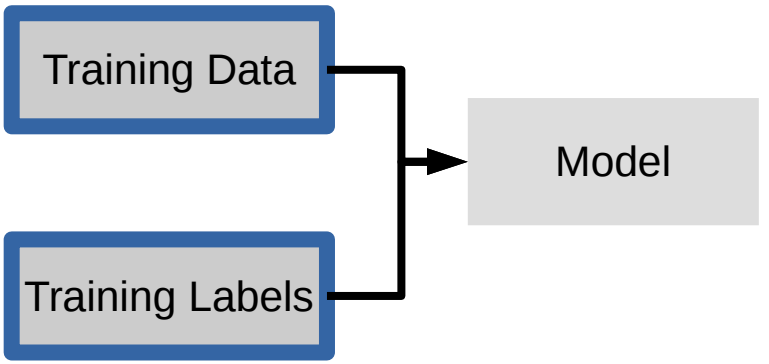


Representing Data



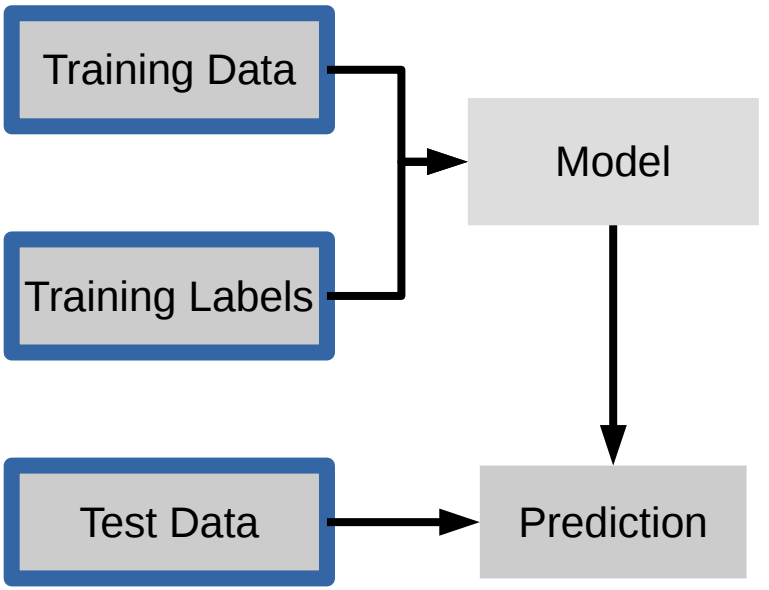
Supervised Machine Learning

```
clf = RandomForestClassifier()

clf.fit(X_train, y_train)
```



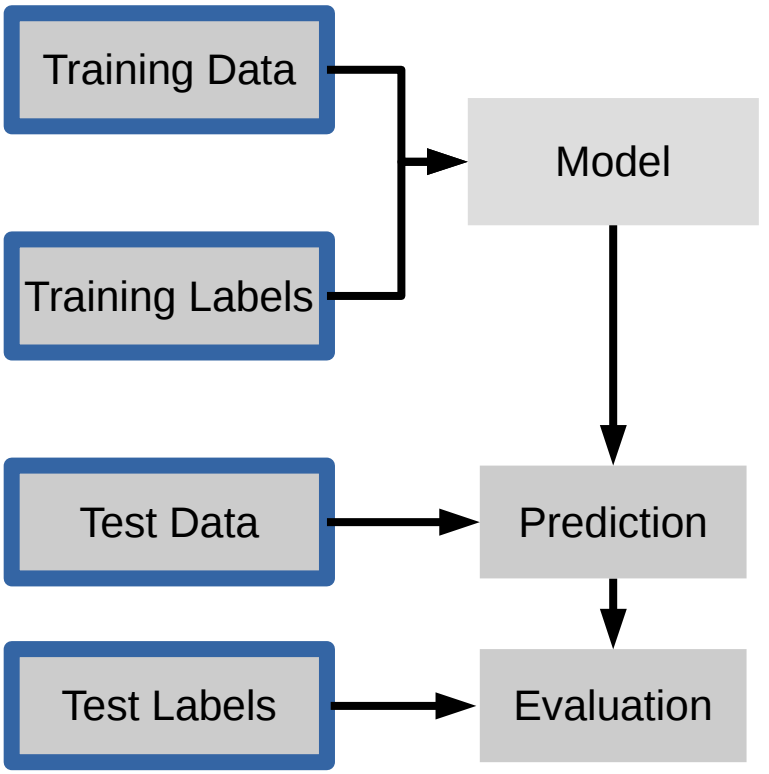
Supervised Machine Learning

```
clf = RandomForestClassifier()

clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)
```



Supervised Machine Learning

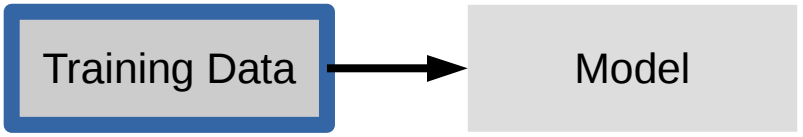
```
clf = RandomForestClassifier()

clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)

clf.score(X_test, y_test)
```



Unsupervised Transformations

```
pca = PCA()
```

```
pca.fit(X_train)
```

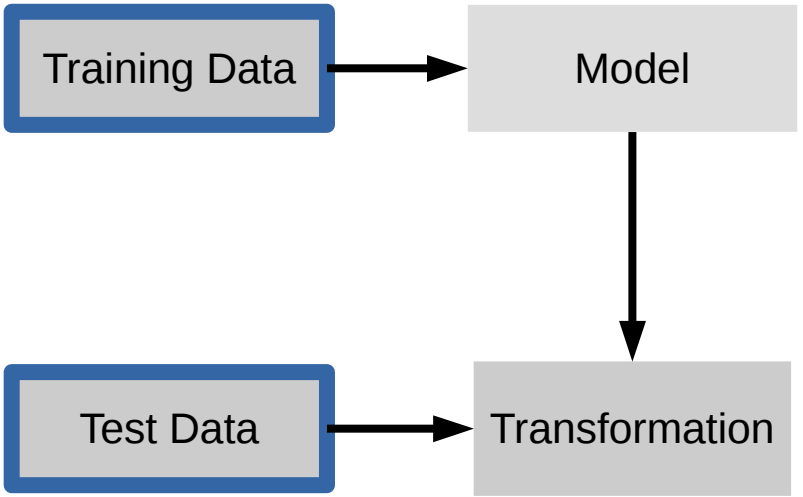


Unsupervised Transformations

```
pca = PCA()
```

```
pca.fit(X_train)
```

```
X_new = pca.transform(X_test)
```



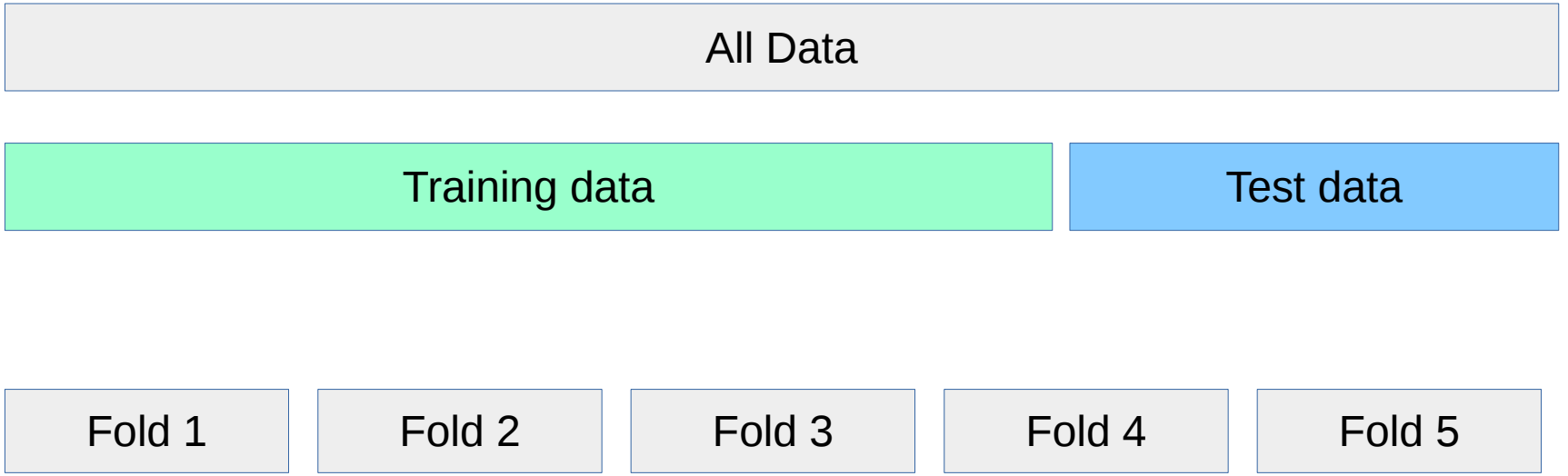
Basic API

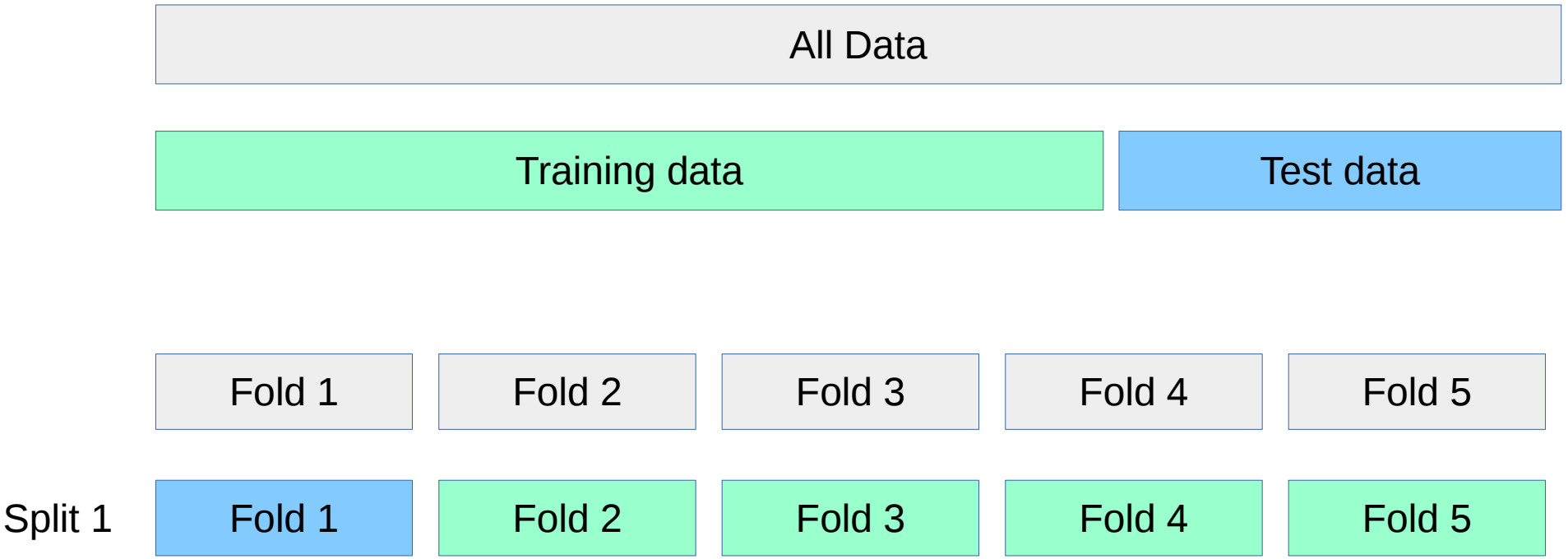
`estimator.fit(X, [y])`

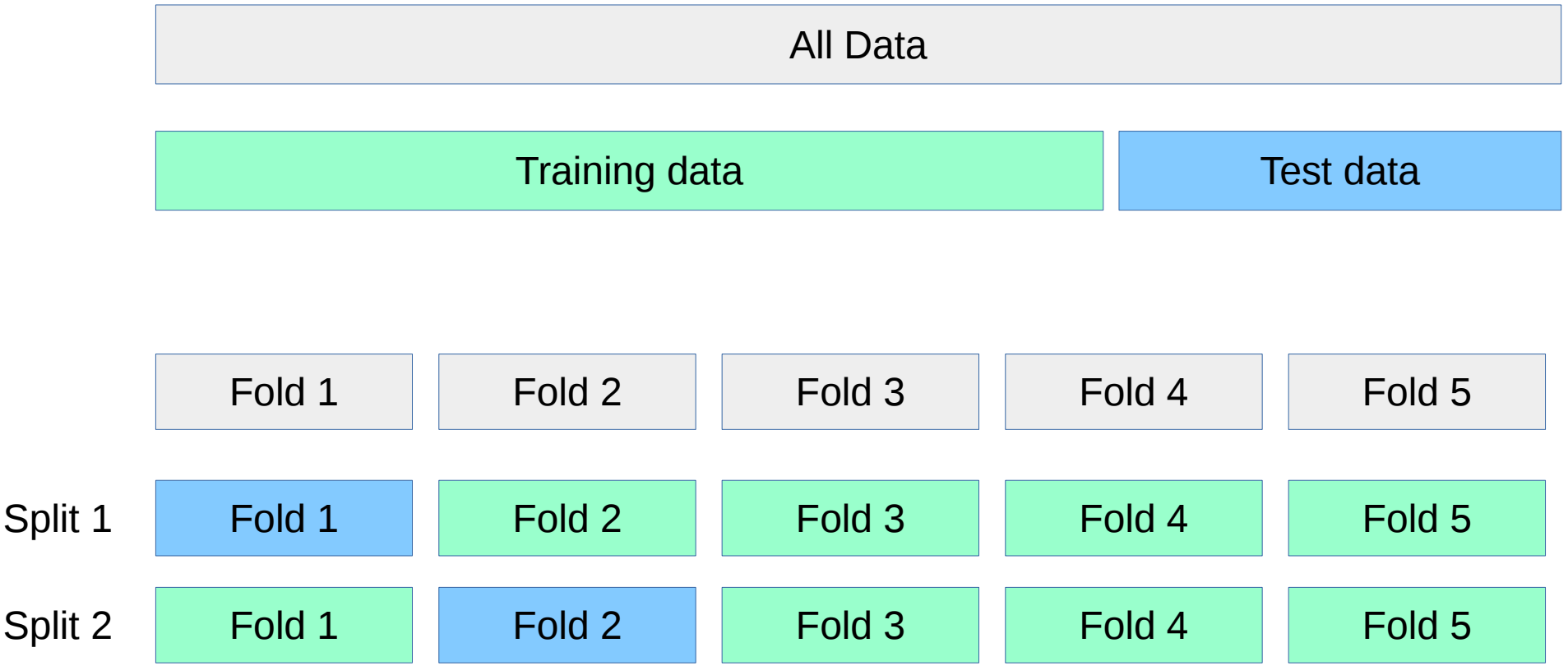
`estimator.predict`	**`estimator.transform`**
Classification	Preprocessing
Regression	Dimensionality reduction
Clustering	Feature selection
	Feature extraction

Model Evaluation and Model Selection


All Data
Training data
Test data


All Data
Training data
Test data
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5


All Data
Training data
Test data
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Split 1
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5

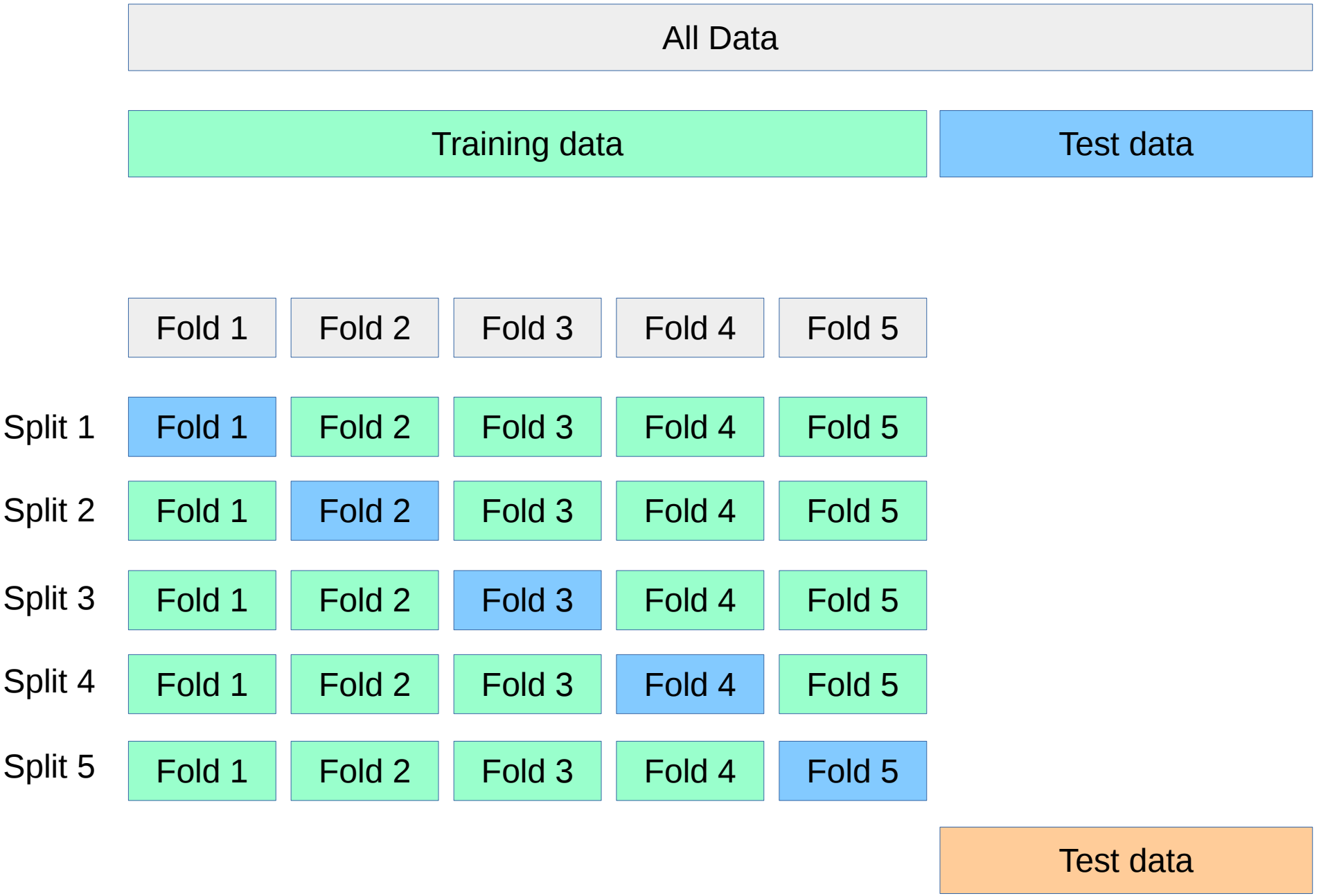

All Data
Training data
Test data
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Split 1
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Split 2
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5

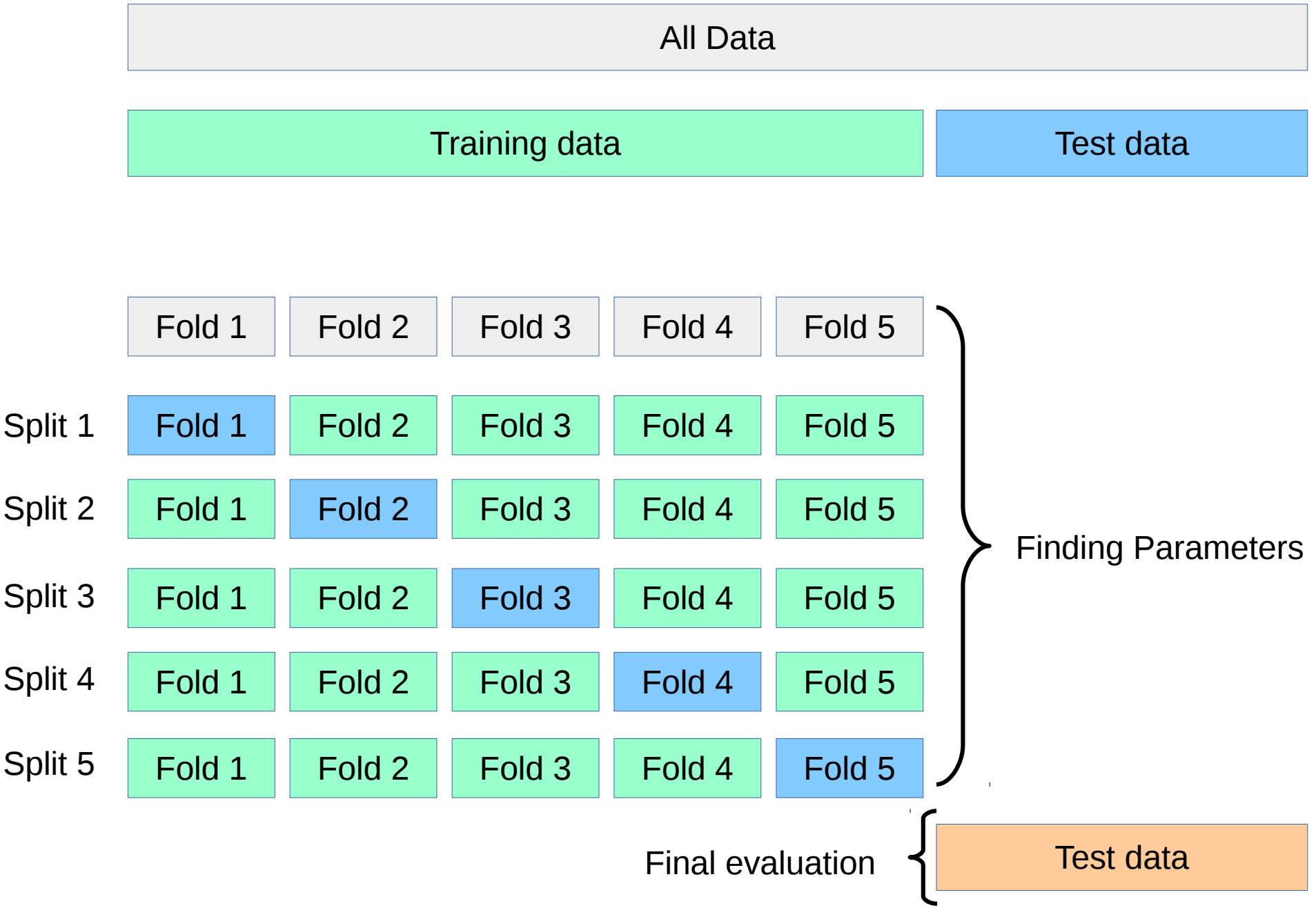

All Data
Training data
Test data
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Split 1
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Split 2
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Split 3
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Split 4
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Split 5
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5

Cross-Validation

```
from sklearn.cross_validation import cross_val_score

scores = cross_val_score(SVC(), X, y, cv=5)
print(scores)

>> [ 0.92  1.    1.    1.    1.  ]
```

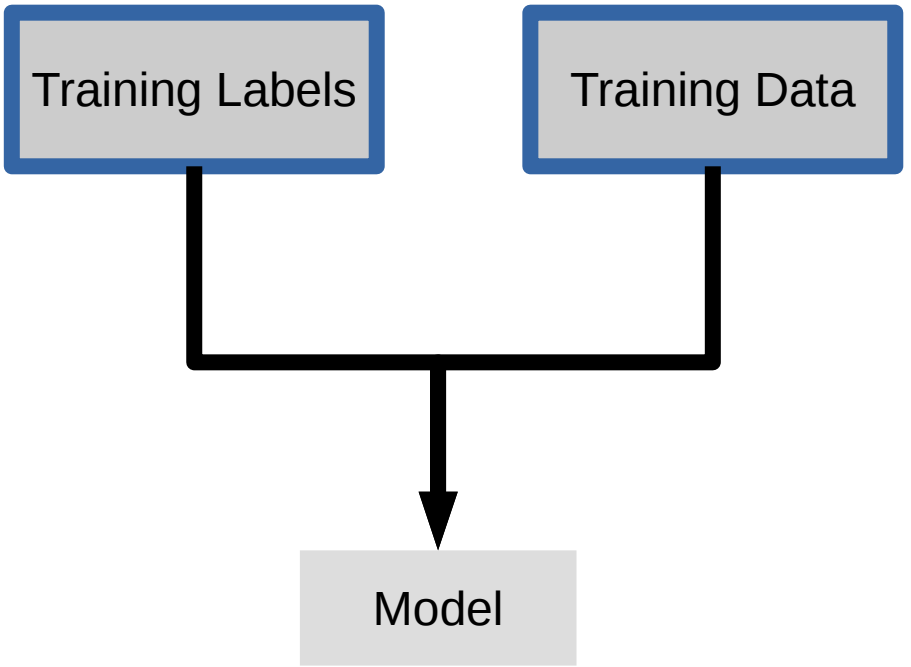

All Data
Training data
Test data

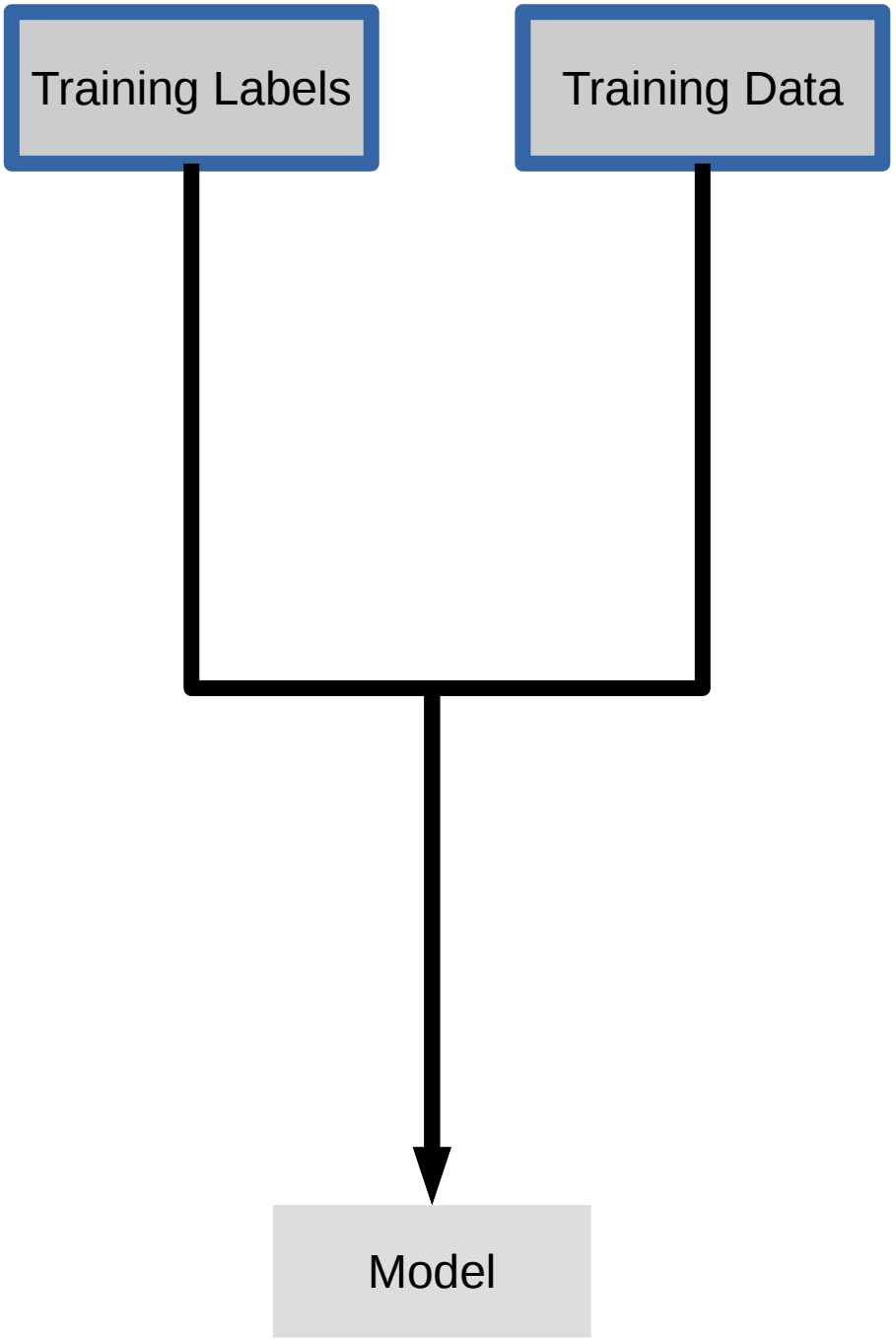

All Data
Training data
Test data
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Split 1
Split 2
Split 3
Split 4
Split 5
Test data

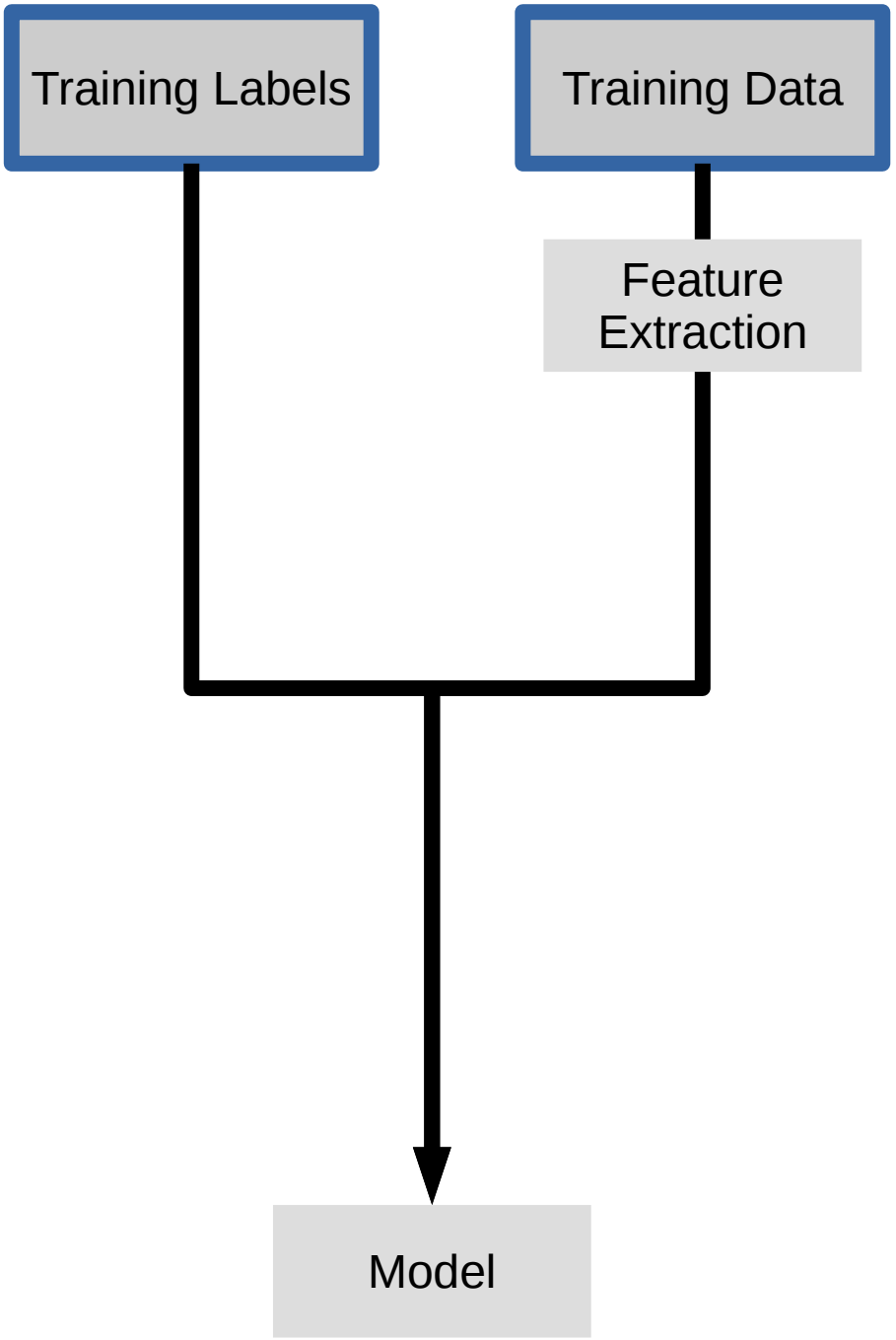

All Data
Training data
Test data
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Split 1
Split 2
Split 3
Split 4
Split 5
Finding Parameters
Final evaluation
Test data

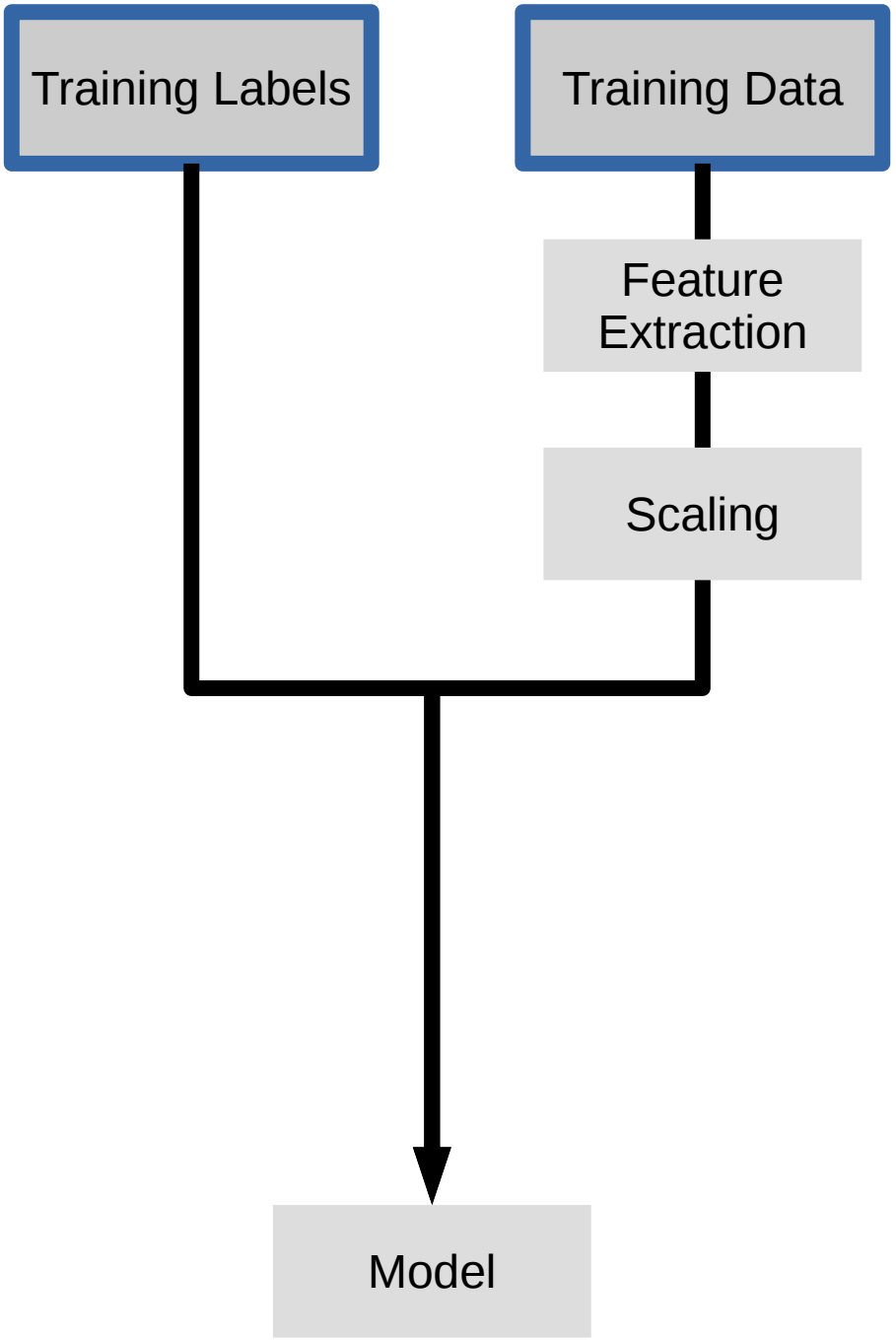
Cross -Validated Grid Search

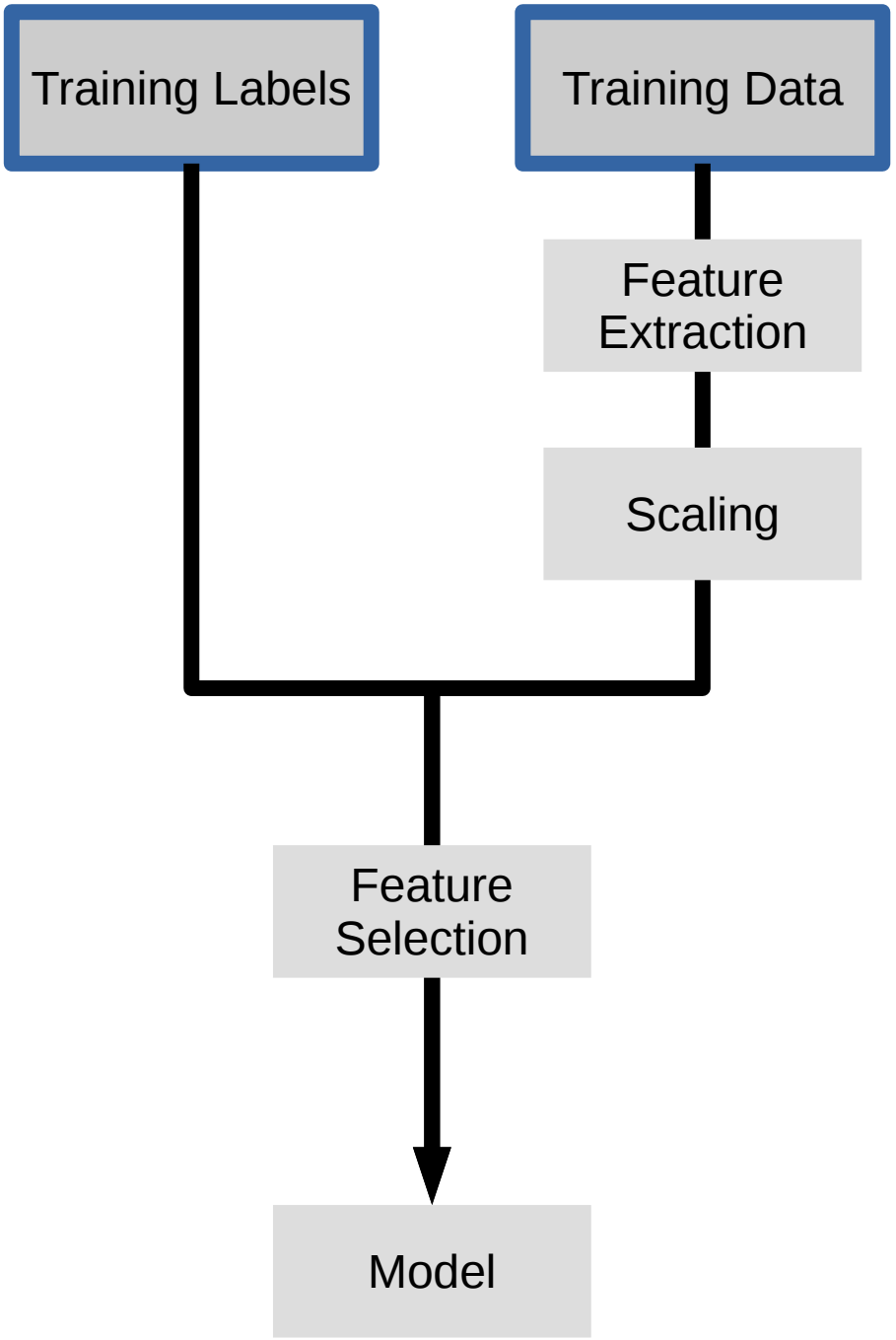
```
from sklearn.grid_search import GridSearchCV
from sklearn.cross_validation import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y)

param_grid = {'C': 10. ** np.arange(-3, 3),
              'gamma': 10. ** np.arange(-3, 3)}
grid = GridSearchCV(SVC(), param_grid=param_grid)
grid.fit(X_train, y_train)
grid.predict(X_test)
grid.score(X_test, y_test)
```

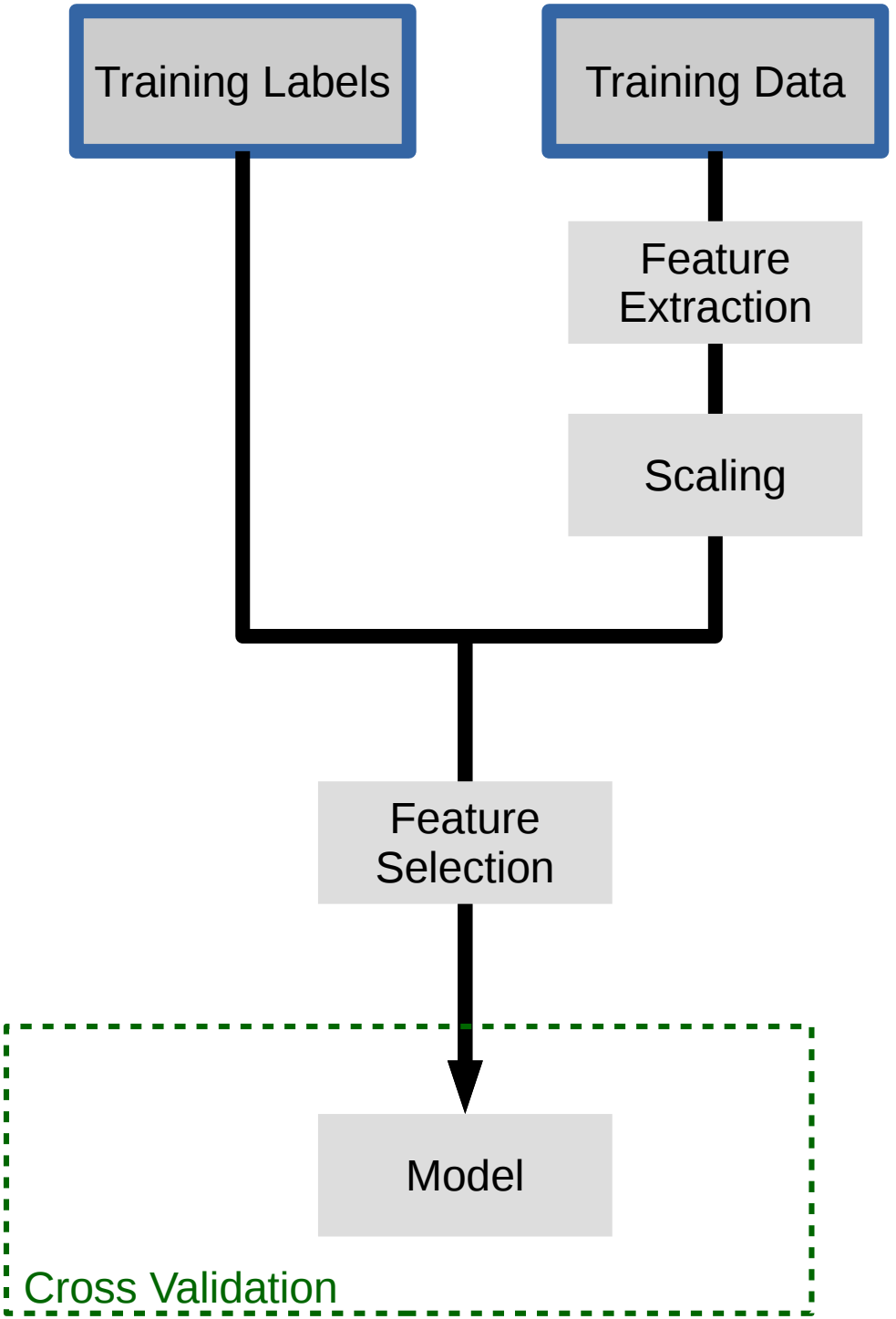

Training Labels
Training Data
Model

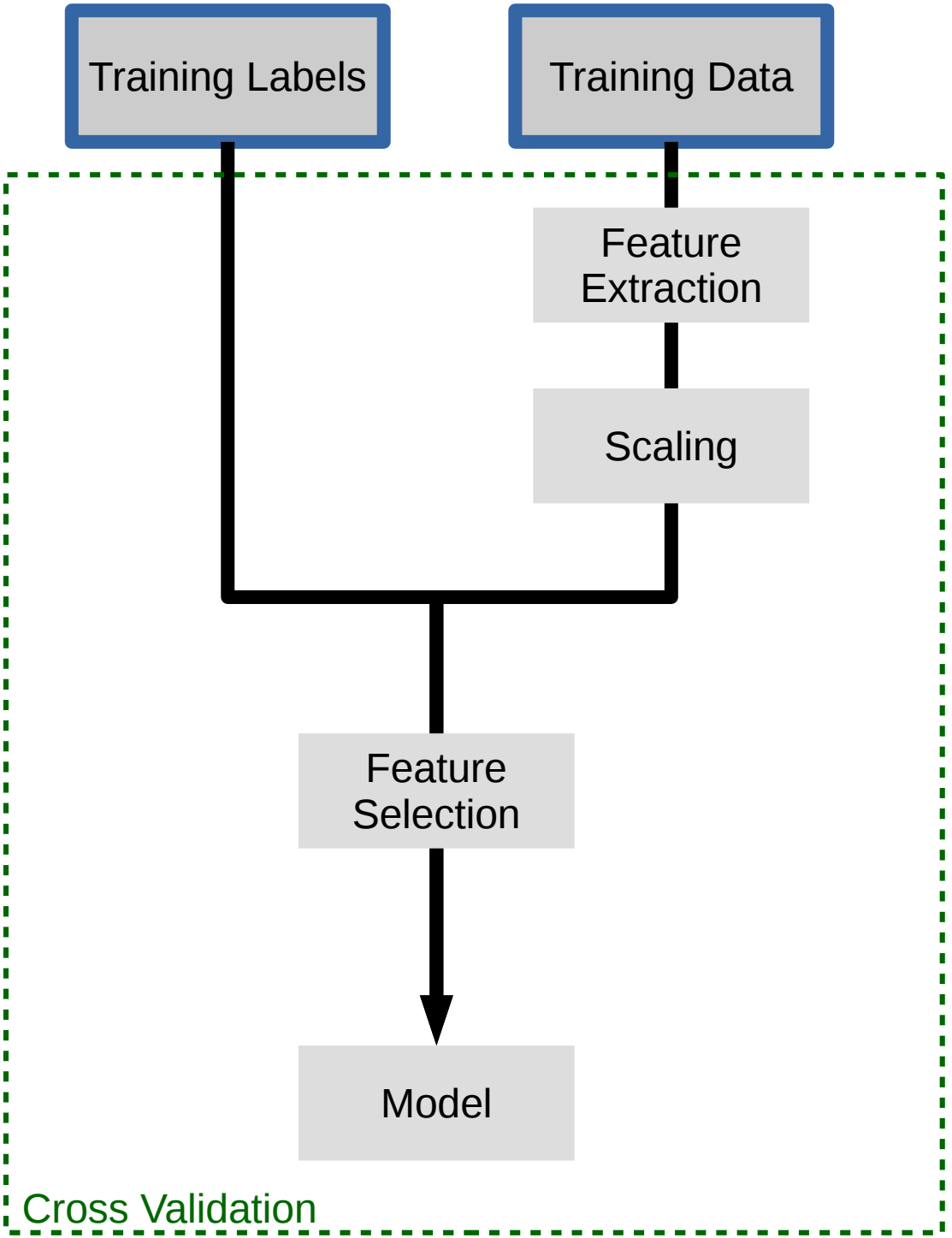

Training Labels
Training Data
Model


Training Labels
Training Data
Feature Extraction
Model


Training Labels
Training Data
Feature Extraction
Scaling
Model

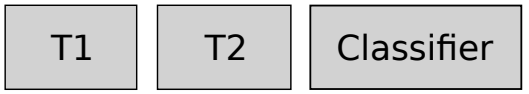

Training Labels
Training Data
Feature Extraction
Scaling
Feature Selection
Model


Training Labels
Training Data
Feature Extraction
Scaling
Feature Selection
Model
Cross Validation

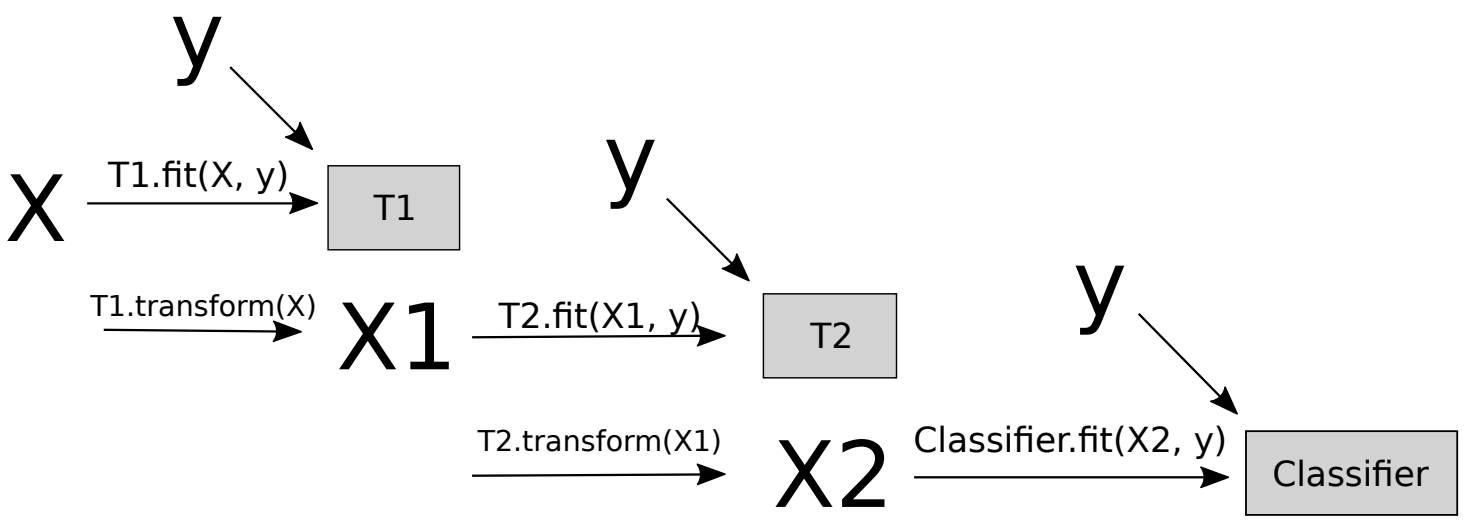

Training Labels
Training Data
Feature Extraction
Scaling
Feature Selection
Model
Cross Validation

Pipelines

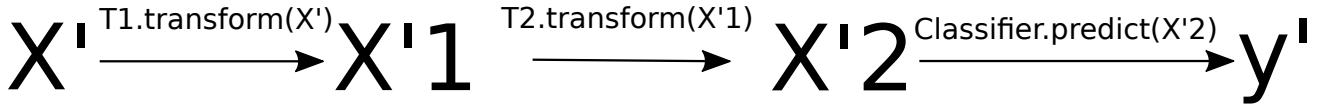
pipe = make_pipeline(T1(), T2(), Classifier())



pipe.fit(X, y)



pipe.predict(X')



Combining Pipelines and Grid Search

```
param_grid = {'svc__C': 10. ** np.arange(-3, 3),
              'svc__gamma': 10. ** np.arange(-3, 3)}

scaler_pipe = make_pipeline(StandardScaler(), SVC())
grid = GridSearchCV(scaler_pipe, param_grid=param_grid, cv=5)
grid.fit(X_train, y_train)
```

Combining Pipelines and Grid Search II

Searching over parameters of the preprocessing step

```
param_grid = {'selectkbest__k': [1, 2, 3, 4],
              'svc__C': 10. ** np.arange(-3, 3),
              'svc__gamma': 10. ** np.arange(-3, 3)}

scaler_pipe = make_pipeline(SelectKBest(), SVC())
grid = GridSearchCV(scaler_pipe, param_grid=param_grid, cv=5)
grid.fit(X_train, y_train)
```

Do cross-validation over all steps jointly.
Keep a separate test set until the very end.

Sample application: Sentiment Analysis

IMDB Movie Reviews Data

`Review:`
One of the worst movies I've ever rented. Sorry it had one of my favorite actors on it (Travolta) in a nonsense role. In fact, anything made sense in this movie.

Who can say there was true love between Eddy and Maureen? Don't you remember the beginning of the movie ?

Is she so lovely? Ask her daughters. I don't think so.

`Label:` `negative`

`Training data:` `12500 positive, 12500 negative`

Bag Of Word Representations

`CountVectorizer / TfidfVectorizer`

Bag Of Word Representations

`CountVectorizer / TfidfVectorizer`

`“This is how you get ants.”`

Bag Of Word Representations

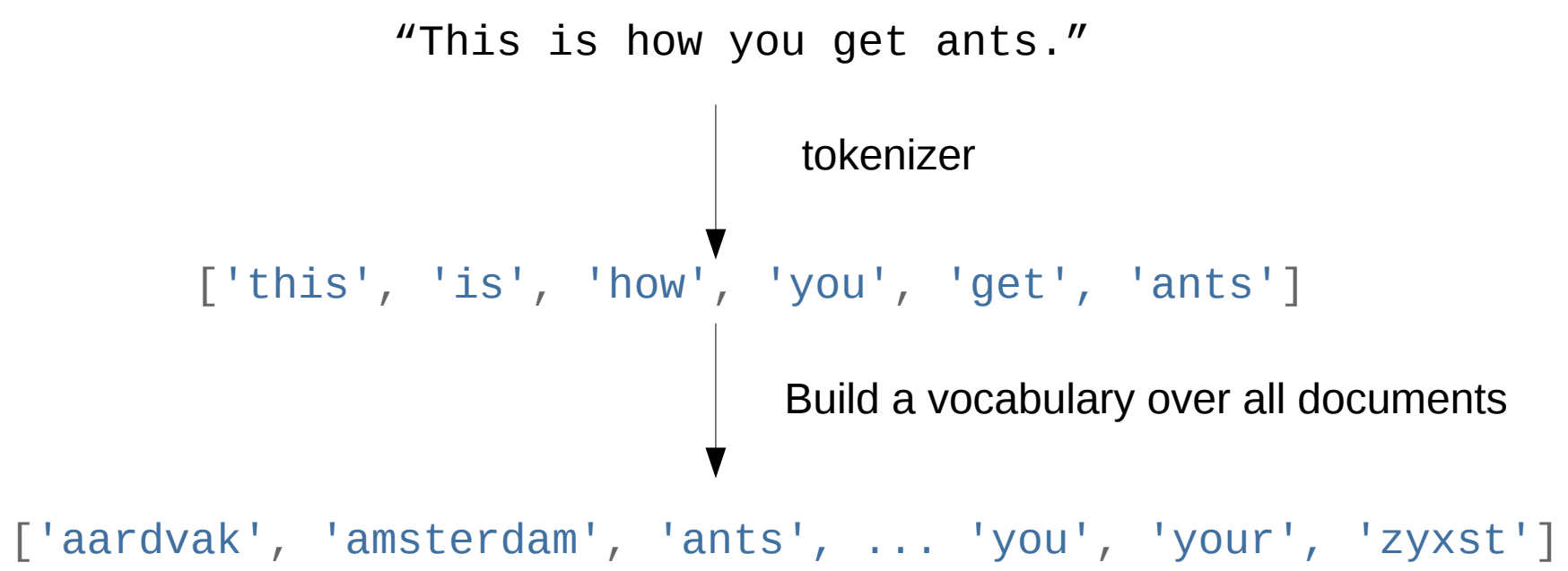
`CountVectorizer / TfidfVectorizer`

`“This is how you get ants.”`


```
['this', 'is', 'how', 'you', 'get', 'ants']
```

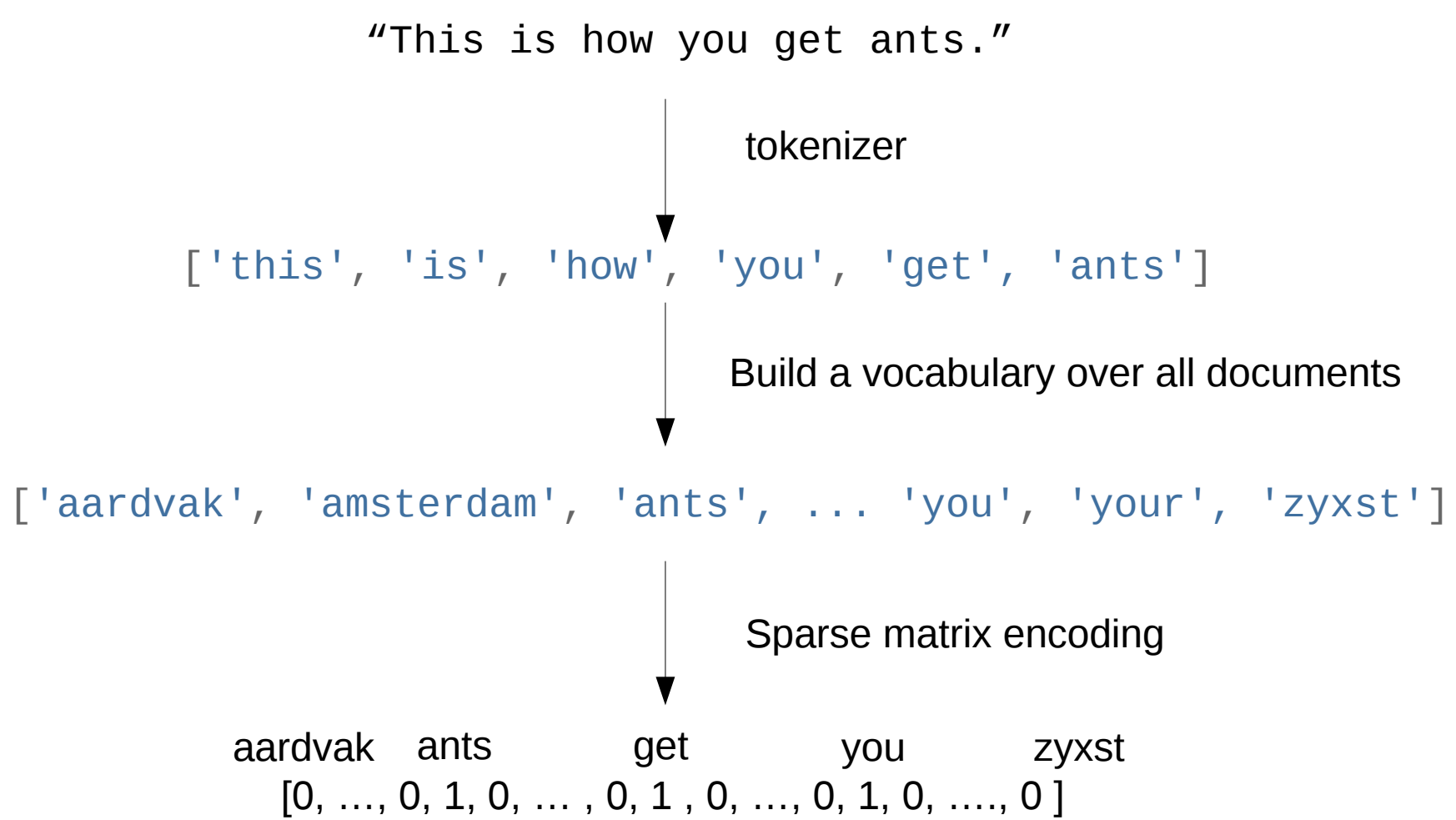
Bag Of Word Representations

`CountVectorizer / TfidfVectorizer`



Bag Of Word Representations

`CountVectorizer / TfidfVectorizer`



N-grams (unigrams and bigrams)

`CountVectorizer` / `TfidfVectorizer`

N-grams (unigrams and bigrams)

`CountVectorizer / TfidfVectorizer`

`“This is how you get ants.”`

N-grams (unigrams and bigrams)

`CountVectorizer / TfidfVectorizer`

`“This is how you get ants.”`

Unigram tokenizer

```
['this', 'is', 'how', 'you', 'get', 'ants']
```

N-grams (unigrams and bigrams)

`CountVectorizer / TfidfVectorizer`

`“This is how you get ants.”`

Unigram tokenizer

```
['this', 'is', 'how', 'you', 'get', 'ants']
```

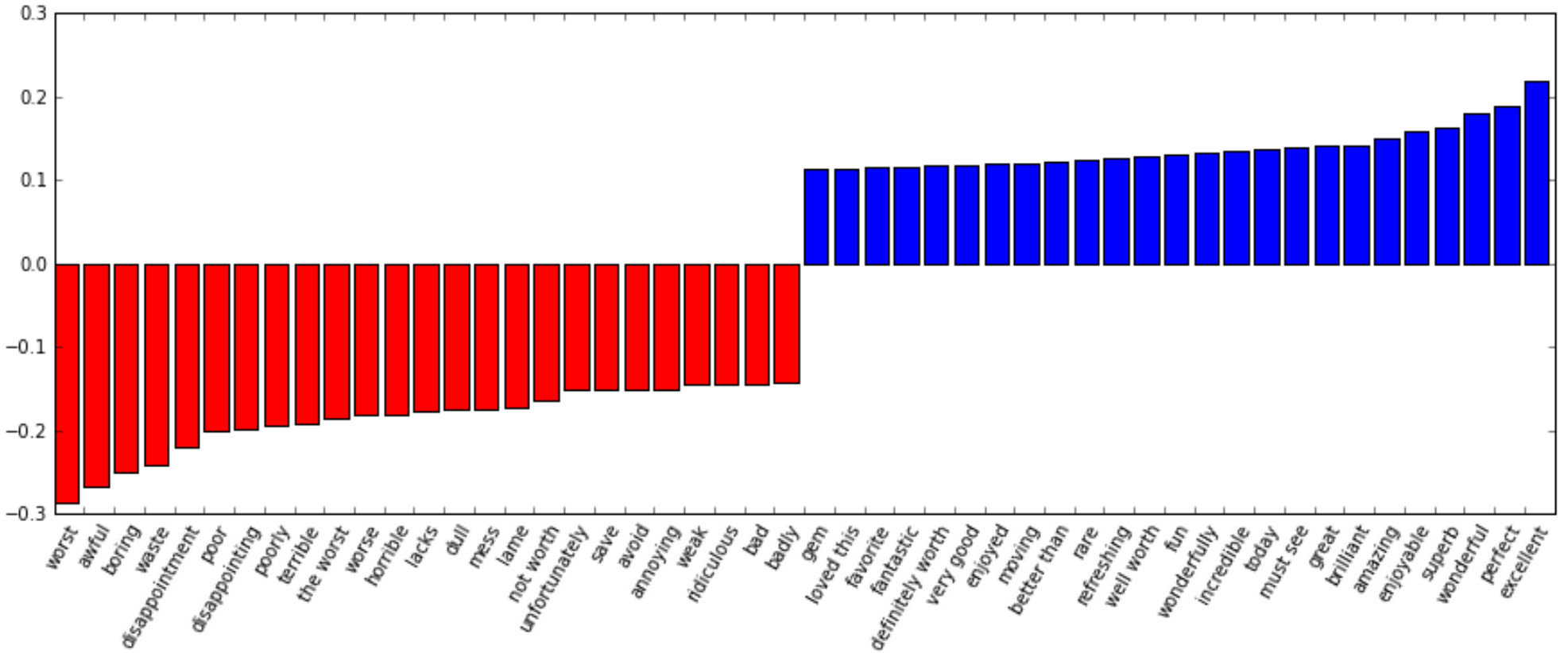
`“This is how you get ants.”`

Bigram tokenizer

```
['this is', 'is how', 'how you', 'you get', 'get ants']
```

```
text_pipe = make_pipeline(CountVectorizer(), LinearSVC())
text_pipe.fit(text_train, y_train)
text_pipe.score(text_test, y_test)
```

```
text_pipe = make_pipeline(CountVectorizer(), LinearSVC())
text_pipe.fit(text_train, y_train)
text_pipe.score(text_test, y_test)
```



Scaling Up

Three regimes of data

- Fits in RAM
- Fits on a Hard Drive
- Doesn't fit on a single PC

Three regimes of data

- Fits in RAM (up to 256 GB?)
- Fits on a Hard Drive (up to 6TB?)
- Doesn't fit on a single PC

Nobody ever got fired for using Hadoop on a cluster

Antony Rowstron, Dushyanth Narayanan, Austin Donnelly, Greg O'Shea, and Andrew Douglas

10 April 2012

"256Gb ought to be enough for anybody."

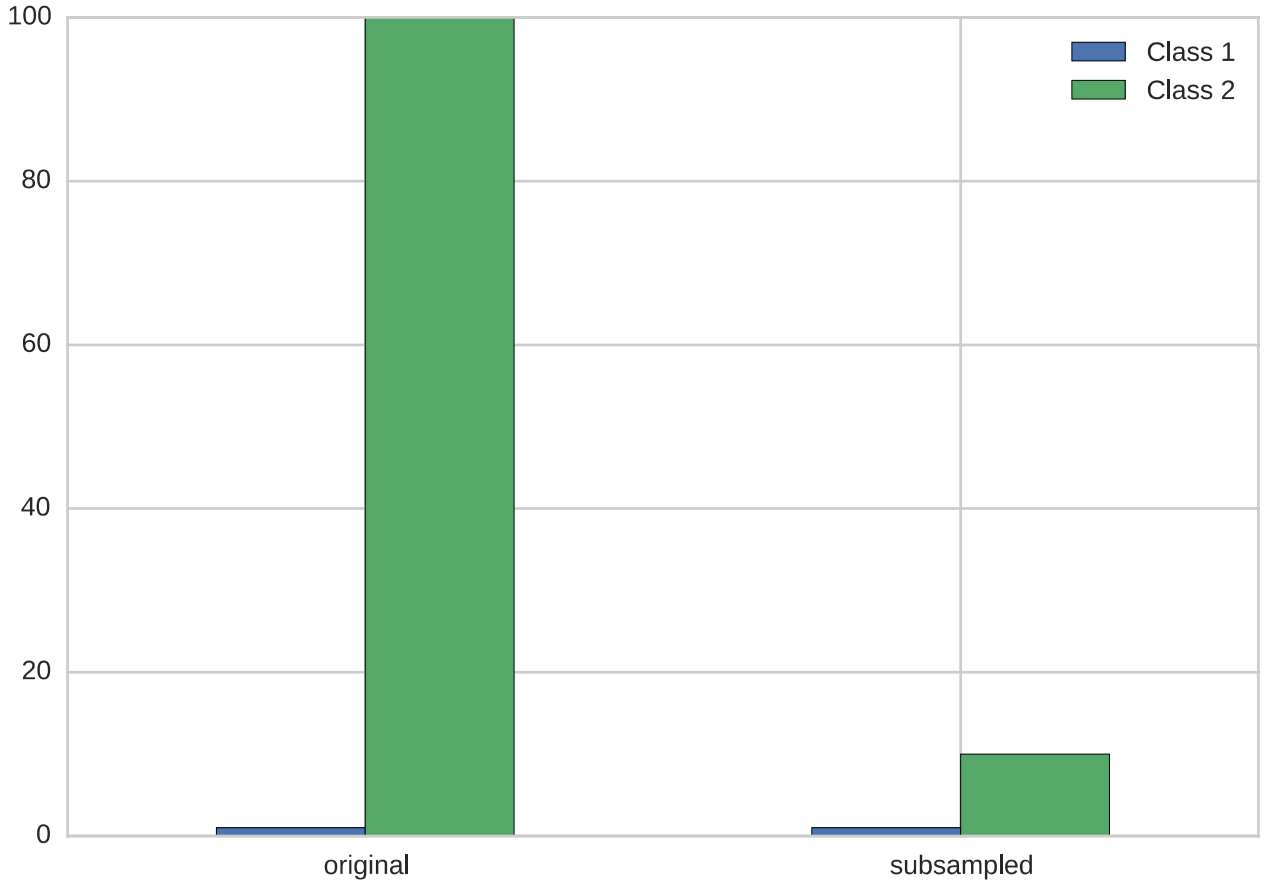
- me

"256Gb ought to be enough for anybody."
- me

(for machine learning)

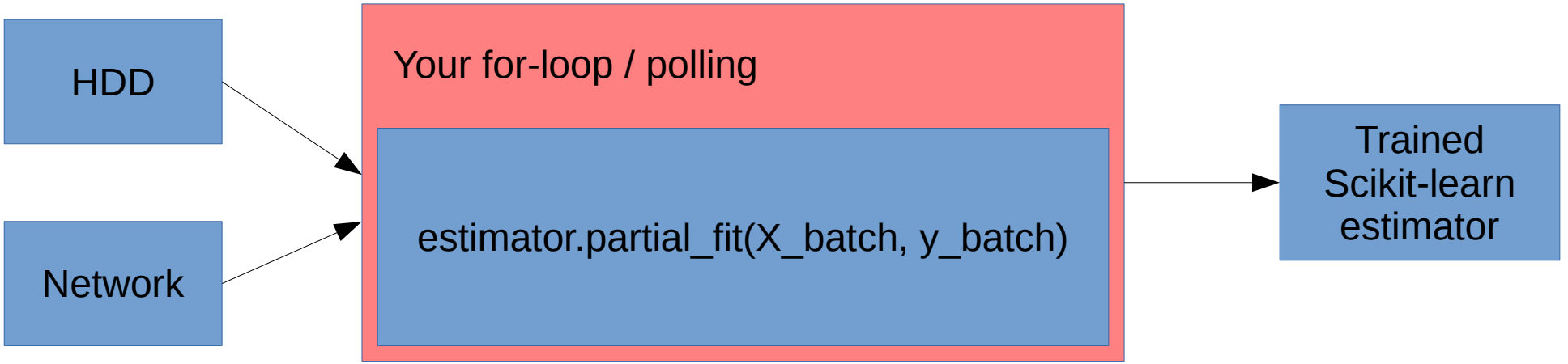
Subsample!

Subsample!



Out of core: The scikit-learn way

The Partial Fit Interface

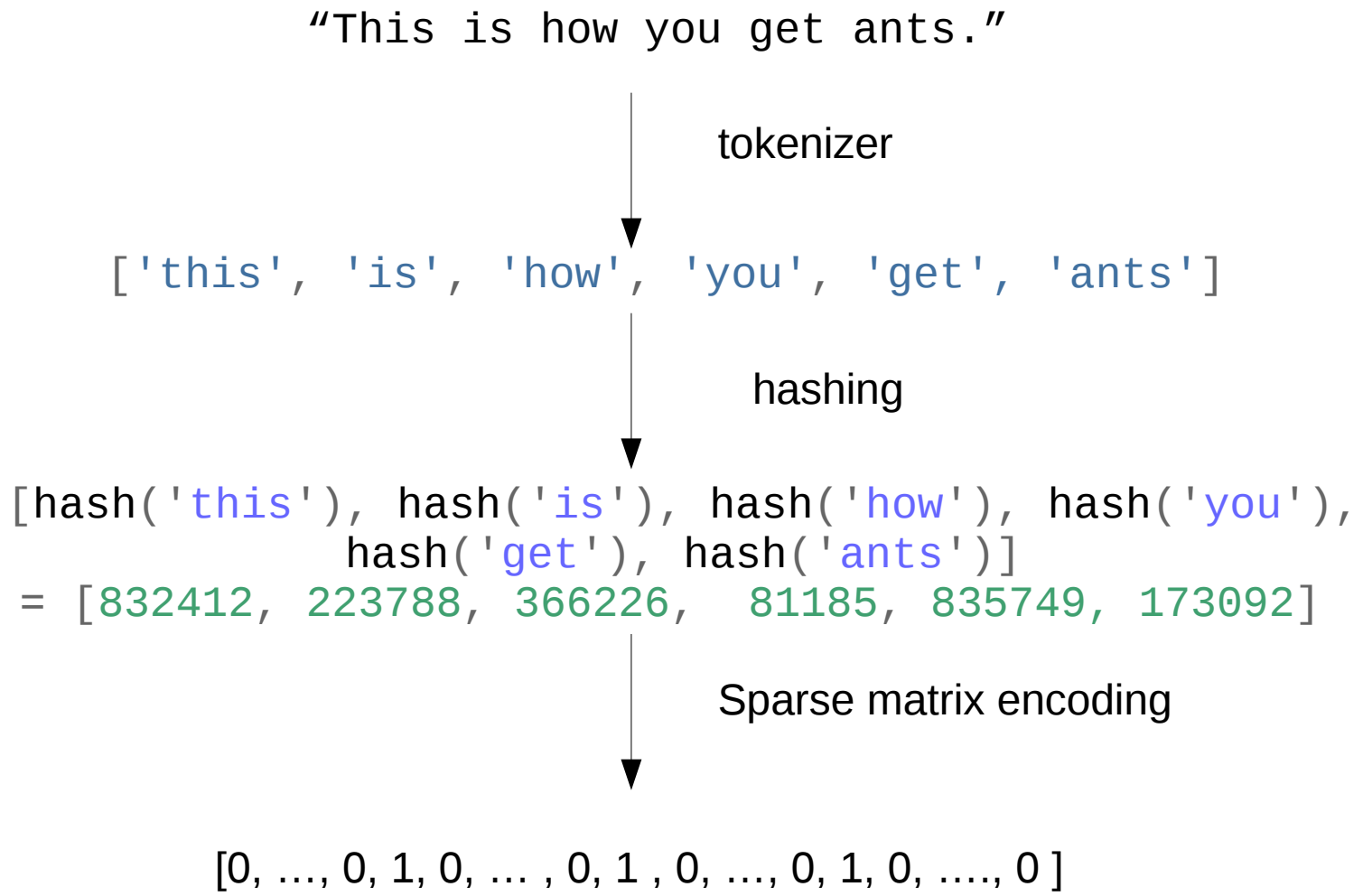


Supported Algorithms

- SGDClassifier/Regressor, Perceptron
- Naive Bayes
- MinibatchKMeans
- Birch
- IncrementalPCA
- MiniBatchDictionaryLearning
- Scalers
- Latent Dirichlet Allocation
- Stateless transformations

Hashing Trick

`HashingVectorizer`



Text Classification: Hashing Trick

```
sgd = SGDClassifier()
hashing_vectorizer = HashingVectorizer()

for batch_name in glob("*.pickle"):
    with open(batch_name) as f:
        text_batch, y_batch = pickle.load(batch_name)

    X_batch = hashing_vectorizer.transform(text_batch)
    sgd.partial_fit(X_batch, y_batch, classes=[0, 1]
```

What's new?

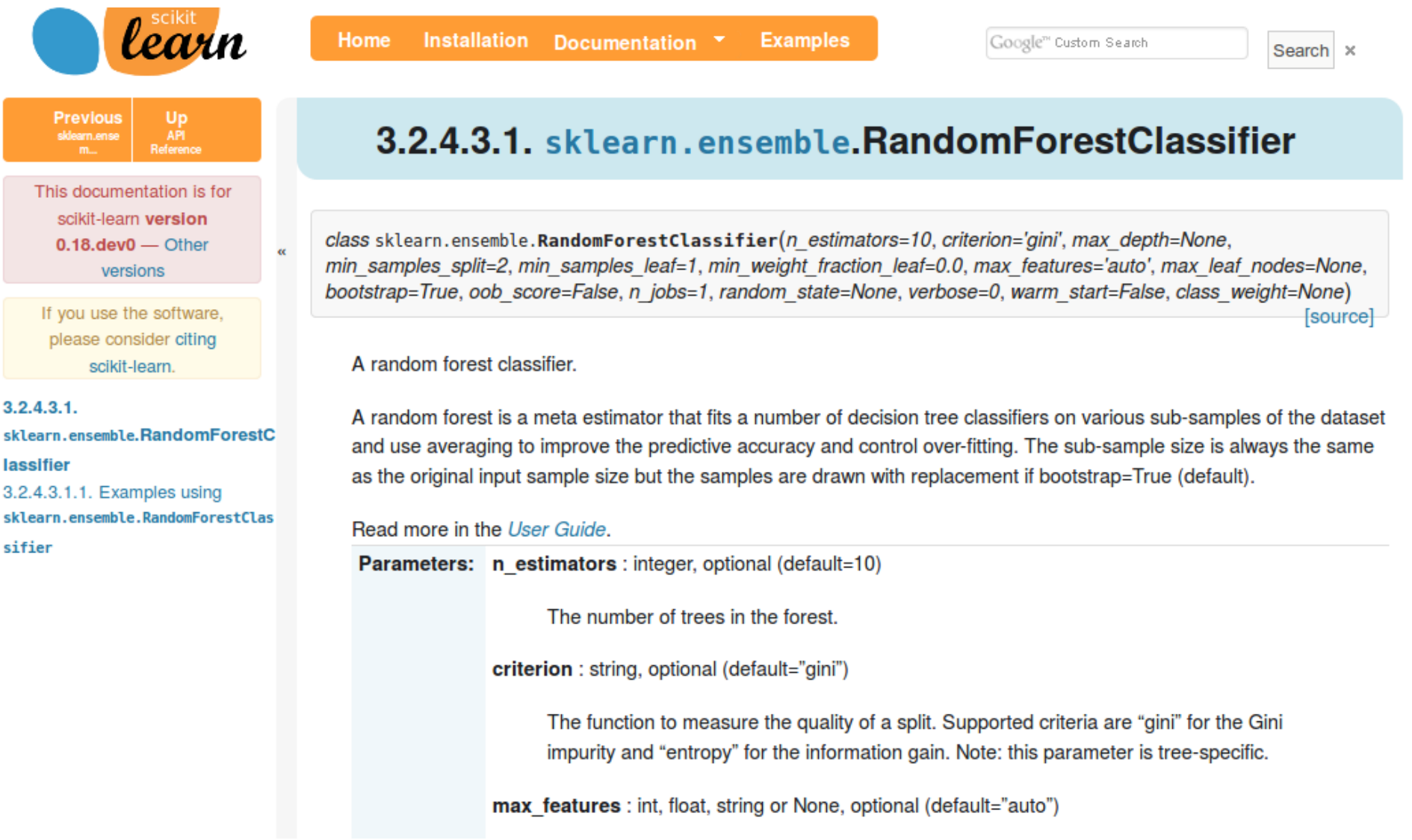
0.17 (stable)

- Latent Dirichlet Allocation
- Faster NMF
- Faster T-SNE
- FunctionTransformer
- VotingClassifier

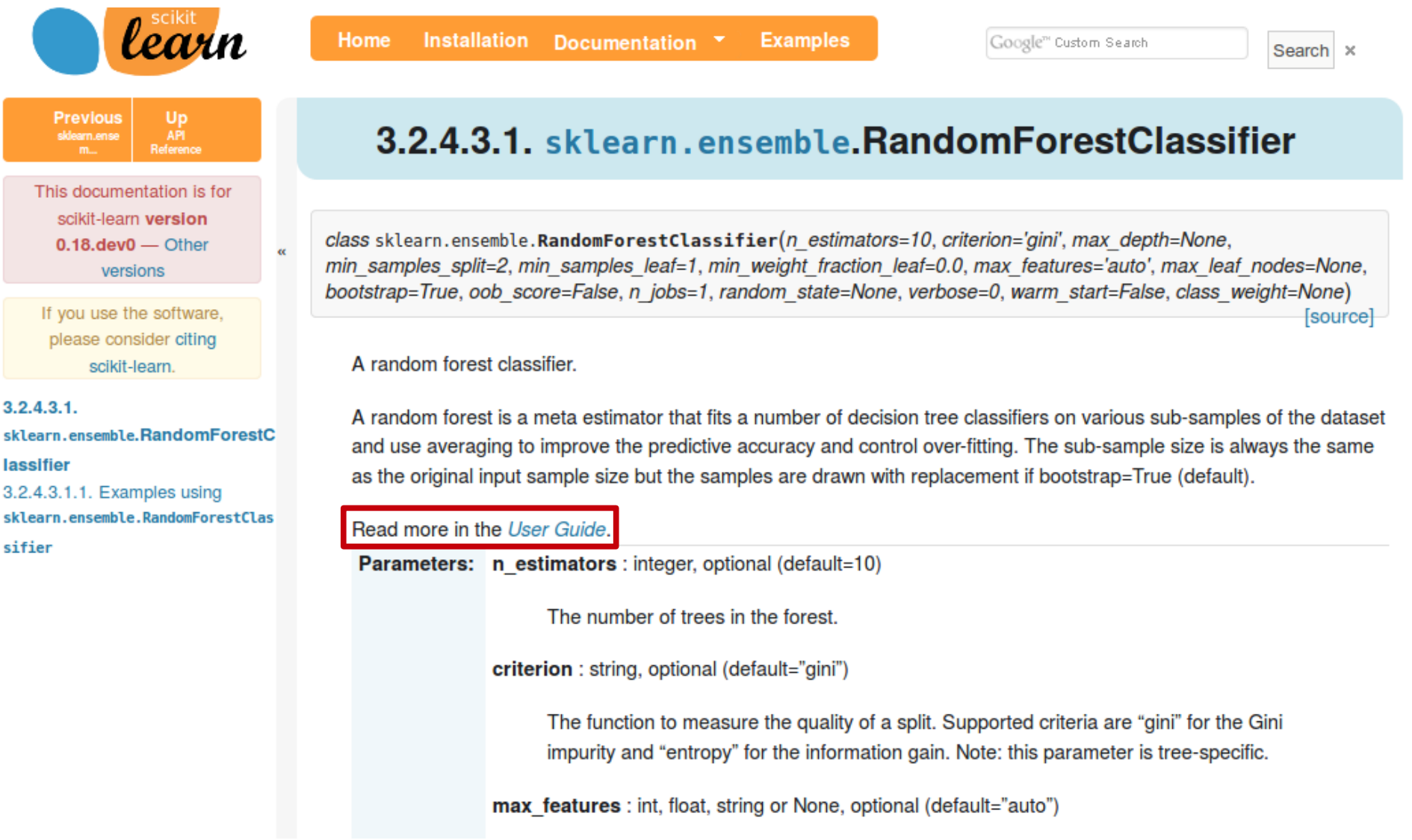
0.18 (development)

- Neural Network
- Gaussian Process rewrite
- Faster PCA

Add Backlinks to Docs



Add Backlinks to Docs



Engineering Scikit-Learn

Goal:
High quality, easy to use machine learning library.

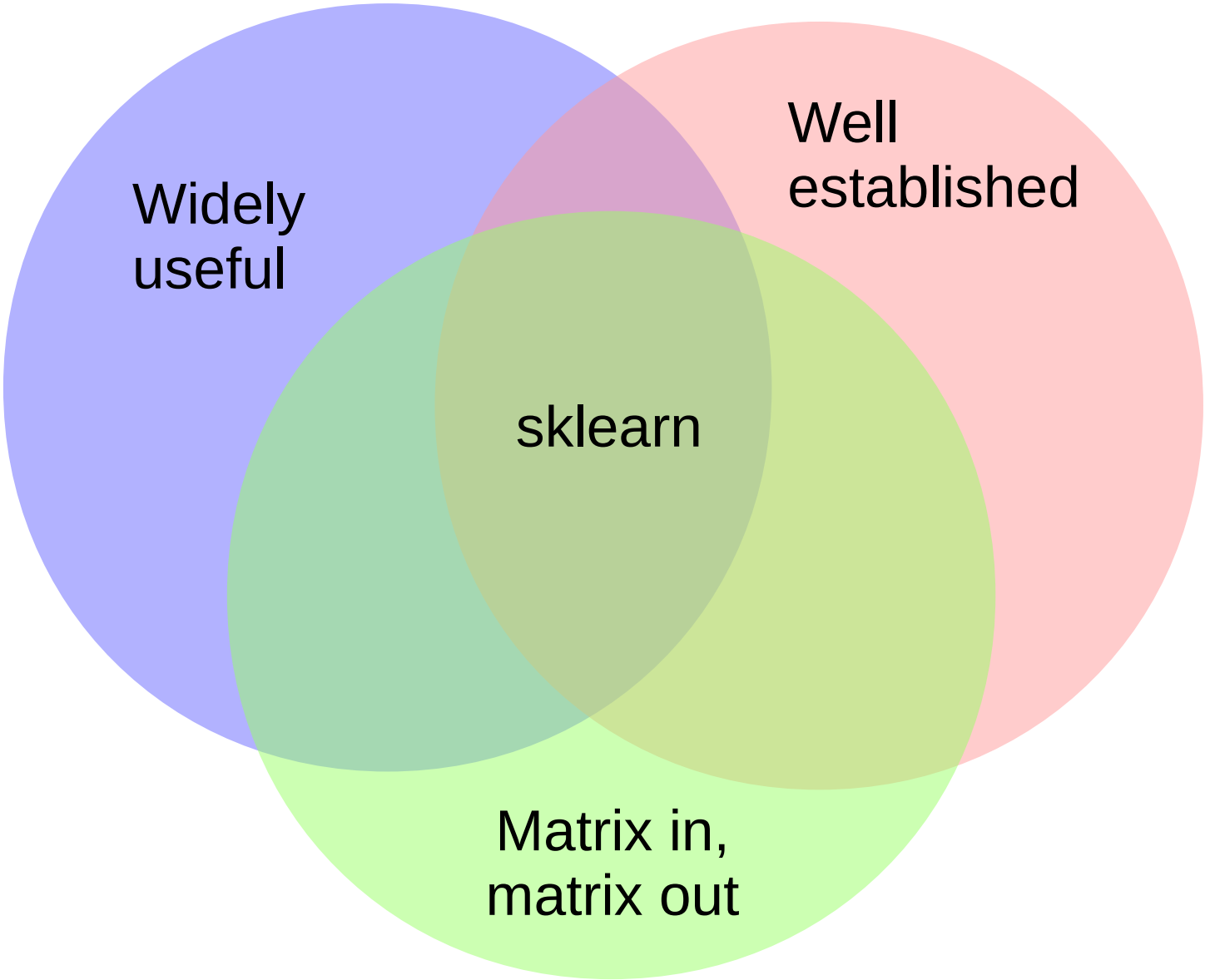
Goal:
High quality, easy to use machine learning library.
Keep it usable, keep it maintainable.

Simple things should be simple, complex things should be possible.

Alan Kay

Methods

Scoping


Widely useful
Well established
sklearn
Matrix in, matrix out

Simplicity

```
est = Est()
est.fit(X_train, y_train)
est.score(X_test, y_test)
```

Consistency

```
grid = GridSearchCV(svm,param_grid)
grid.fit(X_train, y_train)
grid.score(X_test, y_test)
```

Sensible Defaults

Everything is default constructible!

```
for clf in [KneighborsClassifier(),
            SVC(),
            DecisionTreeClassifier(),
            RandomForestClassifier(),
            AdaBoostClassifier(),
            GaussianNB(),
            LDA(),
            QDA()]:
  clf.fit(X_train, y_train)
  print(clf.score(X_test, y_test))
```

Common Tests

```
classifiers = all_estimators(type_filter='classifier')
for name, Classifier in classifiers:
    # test classfiers can handle non-array data
    yield check_classifier_data_not_an_array, name, Classifier
    # test classifiers trained on a single label
    # always return this label
    yield check_classifiers_one_label, name, Classifier
    yield check_classifiers_classes, name, Classifier
    yield check_classifiers_pickle, name, Classifier
    yield check_estimators_partial_fit_n_features, name, Classifier
```

Flat Class Hierarchy, Few Types

- Numpy arrays / sparse matrices
- Estimators
- [Cross-validation objects]
- [Scorers]

No Framework

“This looks frameworkish.” means “try again.”

Avoid Code

- Code rots!
- Hail all code deleters!

Three-Way Documentation

1.9. Ensemble methods

The goal of **ensemble methods** is to combine the predictions of several base estimators built with a given learning algorithm in order to improve generalizability / robustness over a single estimator.

Two families of ensemble methods are usually distinguished:

- In **averaging methods**, the driving principle is to build several estimators independently and then to average their predictions. On average, the combined estimator is usually better than any of the single base estimator because its variance is reduced.

  Examples: *Bagging methods*, *Forests of randomized trees*, ...
- By contrast, in **boosting methods**, base estimators are built sequentially and one tries to reduce the bias of the combined estimator. The motivation is to combine several weak models to produce a powerful ensemble.

  Examples: *AdaBoost*, *Gradient Tree Boosting*, ...

`sklearn.ensemble`.RandomForestClassifier

class `sklearn.ensemble.` **`RandomForestClassifier`** (*n_estimators=10, criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, bootstrap=True, oob_score=False, n_jobs=1, random_state=None, verbose=0, warm_start=False*) [source]

A random forest classifier.

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting.

Parameters:

n_estimators : integer, optional (default=10)

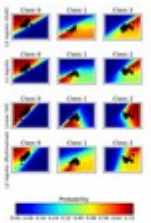
The number of trees in the forest.

criterion : string, optional (default="gini")

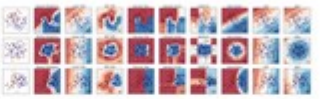
The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain. Note: this parameter is tree-specific.

Examples

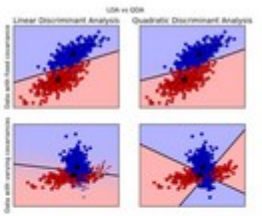
Recognizing hand-written digits



Plot classification probability



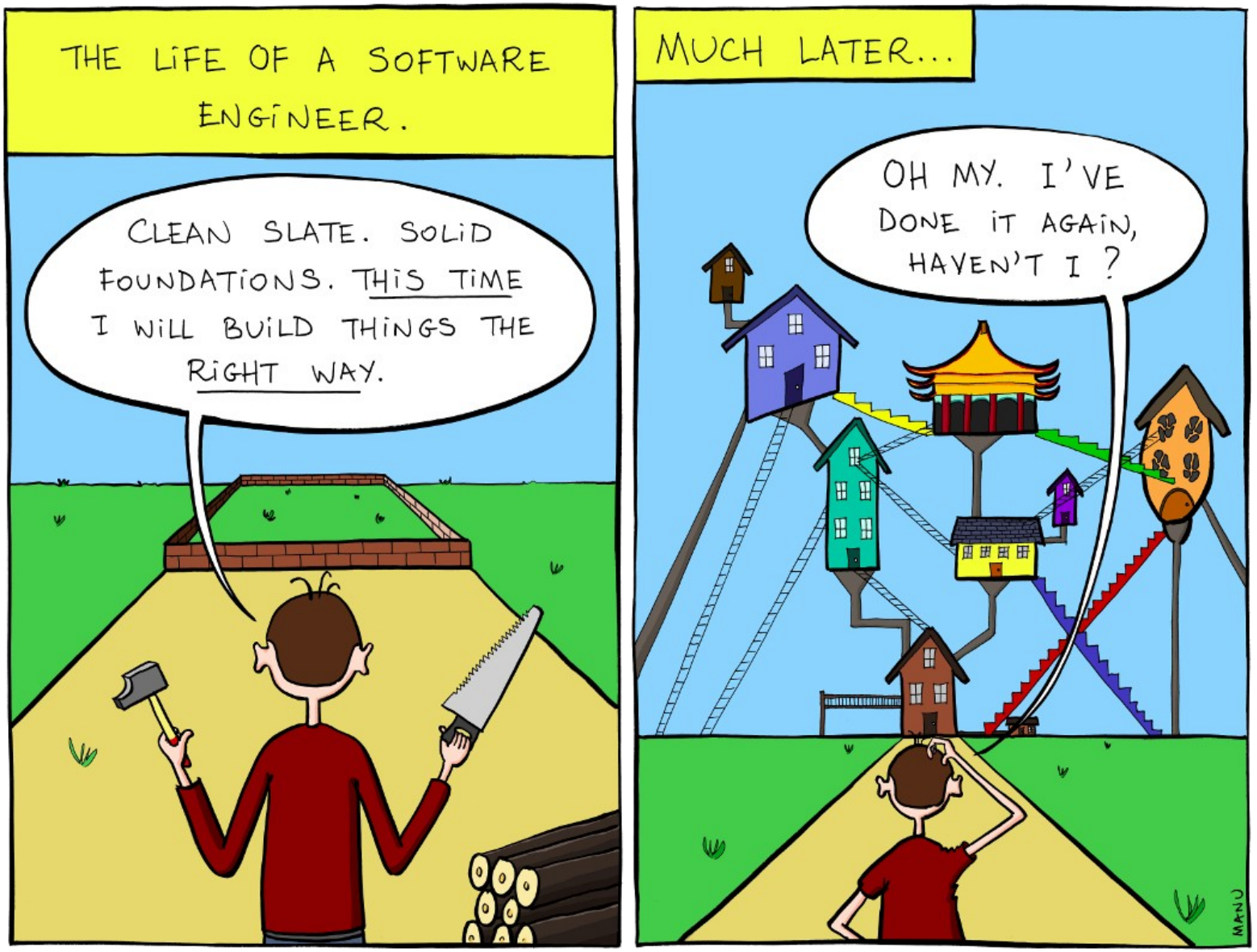
Classifier comparison



Linear and Quadratic Discriminant Analysis with confidence ellipsoid

Challenges

Feature Creep



Two Language Problem

Backward compatibility

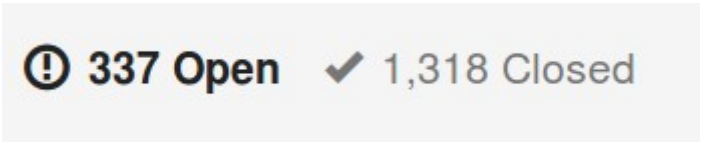
```
from sklearn.cross_validation import Bootstrap
Bootstrap(10)

sklearn/cross_validation.py:685:
DeprecationWarning: Bootstrap will no longer
be supported as a cross-validation method as of
version 0.15 and will be removed in 0.17.
```

Backward compatibility

```
>>> import pickle
>>> s = pickle.dumps(clf)
>>> clf2 = pickle.loads(s)
>>> clf2.predict(X[0])
array([0])
>>> y[0]
0
```

Correctness Testing

Project Size



In a Nutshell, scikit learn...

... has had 17,356 commits made by 424 contributors
representing 433,767 lines of code

Developer churn

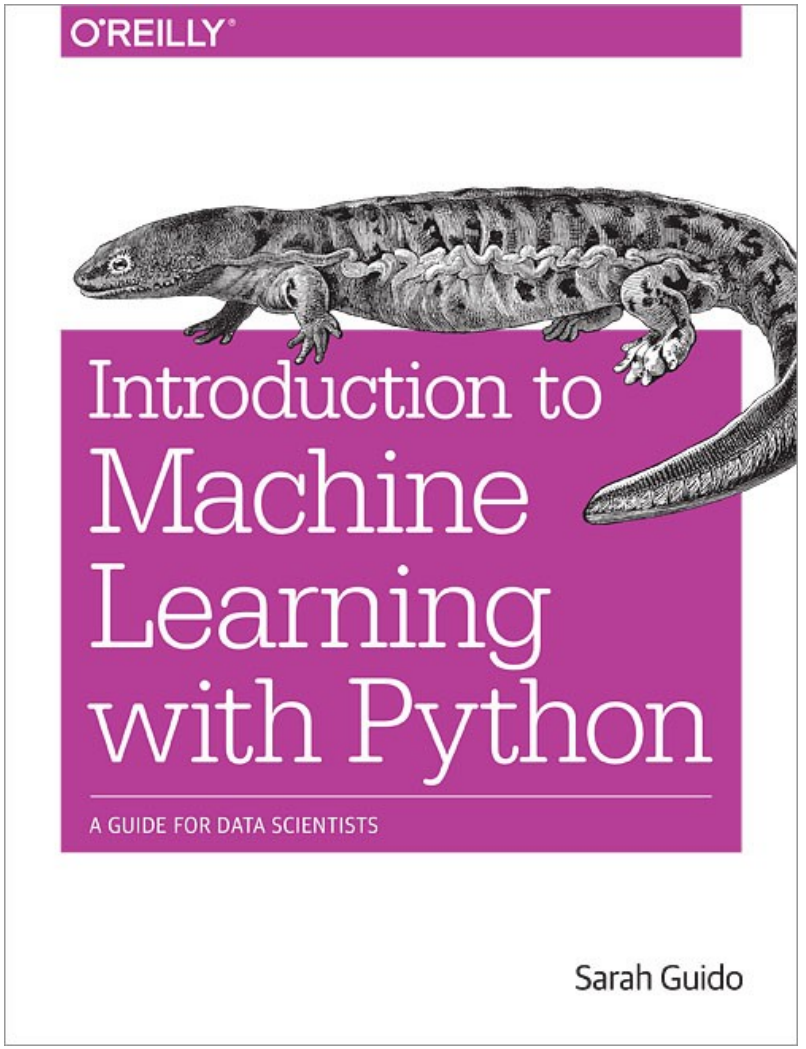
- Two Full time devs (Oliver Grisel & Myself)
- Hundereds of “drive by” contributors

Video Series
Advanced Machine Learning with scikit-learn

50% Off Coupon Code: AUTHD

Video Series
Advanced Machine Learning with scikit-learn

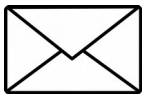
50% Off Coupon Code: AUTHD



Thank you for your attention.


@t3kcit


@amueller


importamueller@gmail.com


http://amueller.github.io